# Step 5 Send commands from IoTHub to Niagara

IoTHub Connector supports cloud to niagara communication via IoTHub C2D messages to control points remotely to do this.

### Send a point action command

1. Go to the Azure IoT Explorer application then in your IoTHub devices then click on the device where you want to send messages. Tehn cloud to device messages.

<u>Home</u> > btibtest > <u>Dev</u>	ices > AHU_01_u0SAYCUHda9VUZ5h5Z7nR > Cloud-to-device message
=	Send message to device
<ul> <li>Device identity</li> <li>Device twin</li> </ul>	Cloud-to-device message 🛈
🖵 Telemetry	Message body O
✓ Direct method	
Cloud-to-device message	
X Module identity	Add timestamp to message body
6 <sup>9</sup> IoT Plug and Play compone	
	🕣 Add custom property 🛛 🖶 Add system property 🗸 🍈 Delete
	Key Value

- 2. By default we use this message template for **POINT\_ACTION** command. by you can use any format that meat your needs. check the connector advanced settings.
  - a. This is the default command template.

b. This is the default connector point action command configuration.

Commands Policy	Single Point Command 🗸
Message Type	<pre>{json('type')}</pre>
Command Set Object	<pre>{json('').escape}</pre>
Command Device Id	<pre>{json('deviceId')}</pre>
Command Point Id	<pre>{json('pointId')}</pre>
Command Action	<pre>{json('payload.action')}</pre>
Command Value	<pre>{json('payload.value')}</pre>
Command Duration	<pre>{json('payload.duration')}</pre>

3. On the left chose Cloud To Device Message.

<u>Home</u> > btibtest > <u>Dev</u>	ices > AHU_01_u0SAYCUHda9VUZ5h5Z7nR > Cloud-to
≡	☑ Send message to device
Device identity	
클 Device twin	Cloud-to-device message ①
☐ Telemetry	Message body ①
> Direct method	"type": "POINT_ACTION", "deviceId": "AHU_01_u0SAYCUHda9VUZ5h5Z7nR", "deviceId": "AHU_01_u0SAYCUHda9VUZ5h5Z7nR",
☑ Cloud-to-device message	"pointld": "Setpoint_v0SAYCUEtoUyS3P0df5kd", "payload": { 
🛠 Module identity	Add timestamp to message body
$\mathcal{S}^{\mathcal{T}}$ loT Plug and Play compone	∧ Properties ①

Then hit Send Message.
 On the niagara you should see the point value applied successfuly.

		Property Sheet	
Network	-	🔘 Setpoint (Numeric	:Writable)
<ul> <li>Discharge air temp</li> </ul>		Facets	units=null,precision=1,min=-inf,max=+inf 📎 🕓
Return air flow sensor		Proxy Ext	null
Discharge air flow			<pre>1111.0 {ok} @ def - {null}</pre>
<ul> <li>W Humidity setpoint</li> <li>Setpoint</li> </ul>	_	— In2	- {null}
Proxy Ext	_	In3	- {null} - {null}
InfoSource		— In5	- {null}
<ul> <li>Return fan command</li> </ul>		<b>—</b> In6	- {null}
-		— In7	- {null}

6. And the new value sent to the cloud.

## Send a send history command

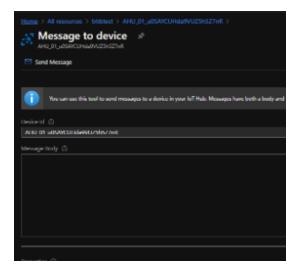
You can pull historical value for any point that has a history associated by sending a SEND\_HISTORY command.

1. Add a history. extension to the point.

\_

	activeBAS/Setpoint			
	Timestamp	Trend Flags	Status	Value
Setpoint	19-Oct-20 1:55:00 PM CEST	{start}	{ok}	1111.0
	19-Oct-20 1:55:10 PM CEST	0	{ok}	1111.0
Proxy Ext	19-Oct-20 1:55:20 PM CEST	0	{ok}	1111.0
InfoSource	19-Oct-20 1:55:30 PM CEST	0	{ok}	1111.0
Milliosource	19-Oct-20 1:55:40 PM CEST	0	{ok}	1111.0
A IoTHubPointExt	19-Oct-20 1:55:50 PM CEST	0	{ok}	1111.0
	19-Oct-20 1:56:00 PM CEST	0	{ok}	1111.0
NumericInterval				

2. Go to the portal and send the command to the device.



- 3. By default we use this message template for send history command. by you can use any format that meat your needs. check the connector advanced settings.
  - a. This is the default command template.

```
{
    "type": "SEND_HISTORY",
    "deviceId": "AHU_01_u0SAYCUHda9VUZ5h5Z7nR",
    "pointId": "Setpoint_v0SAYCUEtoUyS3P0df5kd",
    "payload": {
            "startDate": "2020-10-19T13:15:55.011+02:00",
                "endDate": "2020-10-19T14:15:55.011+02:00"
    }
}
```

b. This is the default connector send history command configuration.

Commands Policy	Single Point Command 👻	
🗎 Message Type	{json('type')}	()
Command Set Object	<pre>{json('').escape}</pre>	()
Command Device Id	<pre>{json('deviceId')}</pre>	0
Command Point Id	<pre>{json('pointId')}</pre>	0
Command Action	<pre>{json('payload.action')}</pre>	0
Command Value	<pre>{json('payload.value')}</pre>	1
Command Duration	<pre>{json('payload.duration')}</pre>	1
📔 Start Date	<pre>{json('payload.startDate')}</pre>	0
📔 End Date	<pre>{json('payload.endDate')}</pre>	0
Delta	<pre>{json('payload.delta')}</pre>	0
📔 Roll Up	<pre>{json('payload.rollup')}</pre>	0

4. And you should see the messages being sent.

<u>Home</u> > btibtest > <u>Devi</u>	ces > AHU_01_u0SAYCUHda9VUZ5h5Z7nR > Telemetry
=	Stop 🗌 Show system properties 📋 Clear events {} Simulation device
Device identity Device twin	Telemetry <sup>©</sup>
🖵 Telemetry	Consumer group O \$Default
> Direct method	Use built-in event hub Yes
Cloud-to-device message	C Receiving events
Module identity 𝒞 IoT Plug and Play compone	<pre>21750 PM 10/19/2020 {     "body": {         "startTimestamp": "2020-10-15T14:15:50.308+02:00",         "endTimestamp": "2020-10-15T14:15:50.308+02:00",         "endTimestamp": "05BNCUREAGYDITH627AR",         "trendTimes": 205BNCUREAGYDITH627AR",         "trendTimes": 205BNCUREAGYSIP0df5kd",         "status: "(0k)",         "status: "(0k)",</pre>

To change the message format check the connector advanced setting then history message template

🗎 History Message Variables	S(startTimestamp) S(endTimestamp) S(pointId) S(deviceId) S(status) S(value) S(trendsFlags) S(count) S(min) S(max) S(max) S(avg) S(sum)
📔 History Message Template	<pre>{     "startTimestamp": "\$(startTimestamp)",     "endTimestamp": "\$(endTimestamp)",     "deviceId": "\$(deviceId)",     "pointId": "\$(pointId)",     "trendsFlags": "\$(trendsFlags)",     "status": "\$(status)",     "value": \$(value),     "count": \$(count),     "min": \$(min).     "</pre>

#### Send ack alarm command

You can ack alarms by sending an ACK\_ALARM command to any alarm recipient device.

- 1. By default we use this message template for ack alarm command. by you can use any format that meat your needs. check the connector advanced settings.
  - a. This is the default command template.

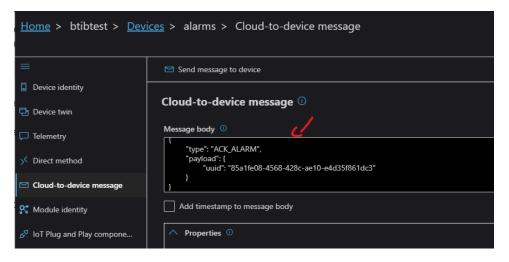
b. This is the default connector send history command configuration.

	"min": \$(min),	*
📔 Alarm Uuid	<pre>{json('payload.uuid')}</pre>	

2. Go the alarms console and pick an unacked alarm id.

Alarm History				
Timestamp	Source State	Ack State	Source	Alar
🜲 19-Oct-20 10:35:36 AM CEST	Offnormal	Unacked	${\it slot:/tesFlex9/points/iotCoreGoogle/GoogleIoTCorePointTestExt}$	Defa
🜲 19-Oct-20 10:35:36 AM CEST	Alarm Rec		· · · · · · · · · · · · · · · · · · ·	
🜲 19-Oct-20 10:35:36 AM CEST	Alaritikee	oru		
🜲 19-Oct-20 10:35:10 AM CEST	Timestamp	19	9-Oct-20 10:35:36 AM CEST	
19-Oct-20 10:35:10 AM CEST	Uuid	85	5alfe08-4568-428c-ael0-e4d35f86ldc3	
19-Oct-20 10:35:10 AM CEST	Source Sta	te Of	ffnormal	
19-001-20 10:55:10 AM CEST	Ack State	Ur	hacked	
🜲 19-Oct-20 10:35:09 AM CEST	Ack Requir	ed ti	rue	

3. Go to device and send the command.



4. Go back to the console and you should see that the alarm has been acked.

Today						
Alarm History						
Timestamp		Source State	Ack State	Source	Alarm Class	
🌲 19-Oct-20 10:35:36 A	M CEST	Offnormal	Acked	${\it slot:/tesFlex9/points/iotCoreGoogle/GoogleIoTCorePointTestExt}$	Default Aları	
🜲 19-Oct-20 10:35:36 A			Unacked	slot:/tesFlex8/points/iotCoreGoogle/GoogleIoTCorePointTestExt	Default Alar	
🌲 19-Oct-20 10:35:3{	4 Alarn	n Record				
A 19-Oct-20 10:35:10	_					
			19-Oct-20 10:35:36 AM CEST			
鼻 19-Oct-20 10:35:1(	-2010:35:10 Uuid 85alfe08-4568-428c-ae10-e4d35f861dc3					
🜲 19-Oct-20 10:35:1(	Sourc	e State (	Offnormal			

## Next Step

Step 6 Consume Data from IoTHub in Azure